GPU-BASED ORTHORECTIFICATION OF DIGITAL AIRBORNE CAMERA IMAGES IN REAL TIME

U. Thomas *, F. Kurz , D. Rosenbaum, R. Mueller , P. Reinartz

DLR (German Aerospace Center), Remote Sensing Technology Institute, 82234 Wessling, Germany
Ulrike.Thomas@dlr.de

Commission VI, WG I/4

KEY WORDS: Direct georeferencing, orthorectification, real time image processing, distributed system architecture, GPU-computation

ABSTRACT:

The usage of airborne camera systems for near real time applications will increase in the near future. This paper purposes a new hardware/software architecture to establish real time computation of images obtained from the DLR wide area airborne 3K-camera system. The main applications of our system are e.g. to monitor automotive traffic, to determine the workload of public road networks during mass events, or to obtain a survey of damages in disaster areas in real time. Therefore, many different image processing tasks have to be executed in real time. Orthorectification of images is necessary prior to all other processing tasks, e.g. before mapping data from street data bases into images or before tracking of vehicles. Nowadays, the calculation becomes possible due to fast graphic processing units (GPU) and with the support of a distributed real time system. In order to achieve real time image processing, we suggest a GPU-based algorithm for image orthorectification. The GPS/IMU-system provides the position and orientation of the aircraft with 128Hz quite accurately. Assuming synchronized measurements with the camera system and given camera calibration, direct orthorectification is implemented using OpenGL. Therewith, we are able to process high resolution images consisting of *16 MPixels* with a frame rate of *3 Hz*. This paper describes the implementation of the real time algorithm and gives first results.

1. INTRODUCTION

Real time processing of imagery airborne data will be very important in the near future. For automatic traffic monitoring, for supporting rescue and security forces, and also for obtaining surveys in disaster scenarios or mass events, an airborne real time image processing system is required. Recently, the 3K-Camera system was developed at DLR (Kurz et al. 2007a). The system consists of three off-the-shelf 16 *MPixel* cameras which are mounted on an airborne platform. Two cameras are directed in side view and one camera is directed in nadir. In the near future, an on-board system consisting of a computer network shall perform image processing in near real time. Important data like traffic payload, mosaiked survey images or changes detected due to disasters should be sent to a ground station in near real time, see figure 1. Various tasks need to be performed on-board, because original high resolution images can not be sent to the ground. This is caused by the data rate which is too high for direct downloading via S-Band microwave. Thus, many processes need to be executed at the on-board hardware. Therefore, we currently develop a distributed image processing system. To obtain high flexibility and good transparency, we suggest the usage of a middleware to handle process-to-process communication across a PC network. Nowadays several middleware platforms are available, e.g. CORBA, TAO. But all of them leak for the possibility to handle large image data in real time. To achieve image processing in real time, strong demands are made to reduce communication as much as possible. In the first part of the paper this novel architecture is described, where many processes need to be performed on the on-board system. Because most of the processes will take orthorectified images as input, strong demands are made on the performance of the orthorectification algorithm. For this purpose, we suggest to exploit common graphics hardware. Hence, the fourth section of this paper explains a GPU-based orthorectification algorithm. With this algorithm, the orthorectification of 3K-camera images becomes possible in real time. The algorithm is completely implemented in OpenGL (Woo et.al. 1999). First results and computation time measurements emphasize our suggestion to exploit GPUs

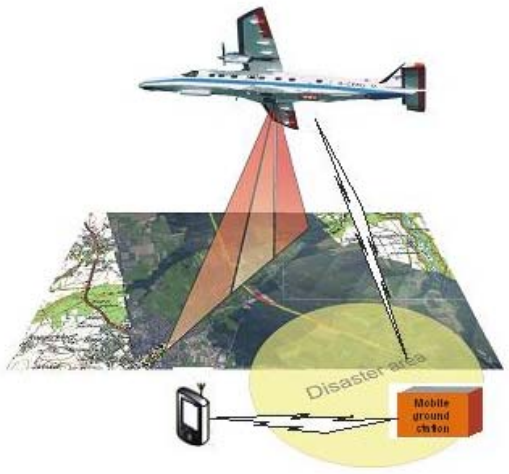

Figure 1: One application scenario for a wide area high resolution airborne camera system.

* Corresponding author.

2. RELATED WORK

Many distributed real time systems have been developed in the last decade. Most of the systems aim hard real time execution where the amount of data to be forwarded between different processes is not very high, (Diethers et al. 2003, Schantz et al. 2003). They are mostly known in the robotics or automation community. In our application, high resolution images have to be processed with a lower frame rate of *3Hz*. The demanding real time image processing system has to cope with data rate of 3 times 16x3 MByte per second (*144 MByte/s* – without compression). Therefore, we introduce a novel software/hardware concept, which is supposed to be able to handle such amount of data. One important image processing task is orthorectification of images. Up to date, analytical implementations have been applied (Mueller et al. 2007). These algorithms burden from the ray casting algorithm, which has to be executed at least for each pixel. Ray intersection is performed by modern graphics hardware many times faster. Furthermore, the analytical computation of surface normal vectors is a time consuming task. One advantage of the analytical solution is the high accuracy, which is useful if distances between sensor and earth surface are high. Here, the GPU-based computation leaks from the logarithmic resolution of depth buffer. Another analytical algorithm for geo-referencing using pattern matching is shown in (Liu 2007). In (Wright et. al 2005) a real time image processing system is developed and also an analytical geo-referencing system is implemented for thermal images. Recently GPU-based computation became very popular for stereo reconstruction.

3. SYSTEM ARCHITECTURE

A brief introduction in our real time image processing system is given, which is currently under development. The DLR 3K-camera system is depicted in figure 2. It is connected via firewire to PC hardware.

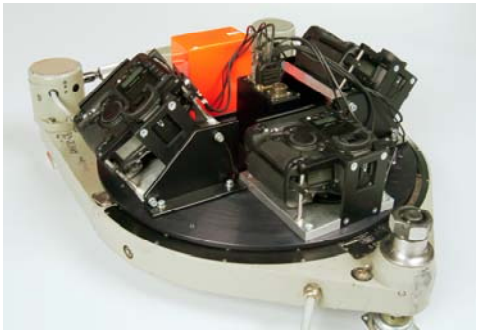

Figure 2: DLR 3K-camera system consisting of three Canon EOS 1Ds Mark II, integrated in a ZEISS aerial camera mount.

The distributed real time system architecture is shown in figure 2. On each computer a middleware is running handling inter-process communication over the network. The middleware also supports the integration of different processes, e.g. if they access the same image date. For this synchronisation the usage of semaphores is comfortable. Thus, the middleware provides the following functionality:

- A name service which takes care of all processes and provides transparency within the network.
- Asynchronous data transmission via message passing (for small data size).
- Synchronised data transmission via shared memory and semaphores (for large data size, e.g. for images).
- Consumer-Producer concept.
- Transparency throughout different computers.

With this concept a layered architecture can be built. Each process consumes data from an upper higher prioritised process and produces data for lower level tasks. For example the ortho-rectification process produces images for the street detection algorithm and itself consumes image data obtained from the sensor directly. This enables a more sophisticated implementation of systems running various processes in real time. One process needs only to know what kind of data it is interested in. Thus, it initialises its messages to be sent to other processes. Also for shared memory usage (e.g. for image data), the process registers its data at the middleware. The middleware establishes correct and monitored data access. Thus, each implemented process establishes communication only with the middleware. The exchange of data through various processes is organised by the middleware. As seen in figure 3, in our system five on-board computers are involved. The first three machines acquire images and process elementary tasks:

- Orthorectification of images.
- Mapping data from a street data base into the orthorectified image.
- Segmentation of streets.

Another computer is available for traffic monitoring, e.g. car detection and car tracking (Kurz et al. 2007b). On this machine various tasks are running. The upper layered task consumes segmented streets and produces traffic data e.g. payload of road network, traffic jam, velocity of vehicles. The machine, which is linked to the microwave system, obtains the data and sends it down via S-Band microwave, which is able to bridge distances over *200 km*. The communication to ground computer networks is done via UDP send. Therewith, real time and near real time applications of an airborne wide-area high resolution camera system becomes possible.

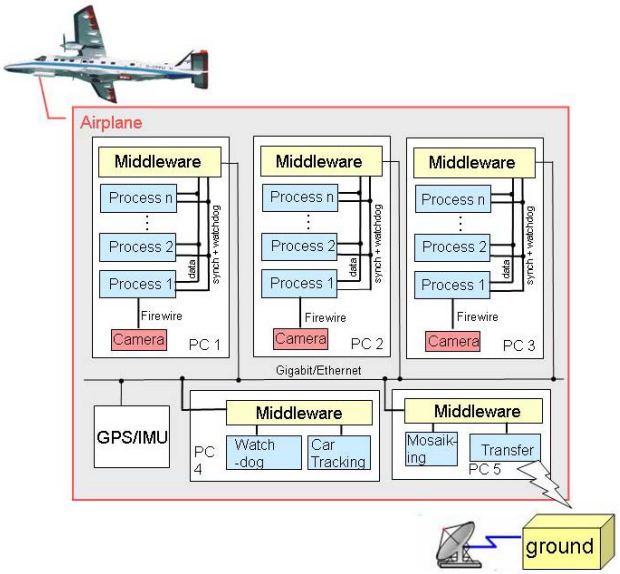

Figure 3: Architecture of our distributed real time on-board system.

4. ORTHORECTIFICATION

Orthorectification of images has to be computed on the on-board hardware. There, GPS/IMU data are available in real time with *128 Hz*. The flight route is planed prior to the flight. In order to rectify images DSMs are necessary. They have to be loaded from a data base prior to flight. Figure 4 illustrates the image acquisition geometry of the DLR 3K-camera system. The tilt angle of the sideward looking camera is approx 35°. Based on the usage of *50 mm* Canon lenses, the dependency between airplane flight height, ground coverage, and pixel size is shown, e.g. the pixel size at a flight height of *1000 m* above ground is *15 cm* and the camera array covers up *2.8 km* in width.

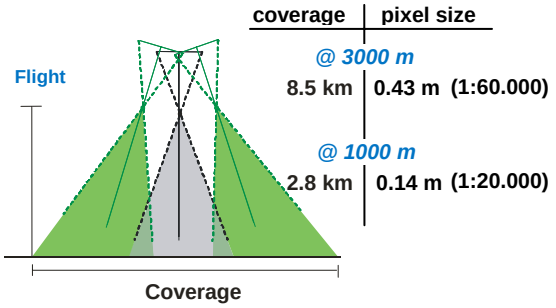

Figure 4: Illustration of the image acquisition geometry. The tilt angle of the sideward looking cameras is approx. 35°.

From this geometry the amount of triangles necessary to cover the ground surface is derived. Table 1 lists the number of triangles necessary for one triple image of the 3K-camera system. For this estimation the RPY-angles of the airplane are assumed to be zero. Of course, the number of triangles may increase according to roll, pitch, and jaw angles as well as the angle between the air plane and the earth surface. Thus, DSMs are loaded on demand into PC storage. For holding the appropriate DSM available, a Kalman-Filter is applied estimating the high probable area and triggering the DSM loading process. Then, the DSM covering this area is triangled as fast as possible and loaded into the GPU. For up-to-date graphics hardware handling such amount of triangles in real time is possible. Some computation times for triangulation are shown in section 5. The triangulation of surfaces will be necessary prior to flight, if DSMs of higher resolution e.g. *2 m.* are applied. For halving the amount of data a Delaunay algorithm can be used, but this algorithm is too slow for real time computation. In this case, the mesh should be generated prior to flight and stored in a binary file.

Flight height	Number of triangles			
	DSM Resolution [*25m*]		DSM Resolution [*2m*]	
	side look	nadir	Side look	nadir
1000 m	1710	1064	274 560	172 800
2000 m	6916	3420	1 099 200	691 200
3000 m	7809	9804	2 473 920	1 555 200

Table 1. Number of triangles necessary according to one triple image of the DLR 3K-camera system (side look and nadir). Assuming a boresight angle of 35°.

Fig. 5 illustrates the complete virtual scene necessary for GPU-based orthorectification. All transformations and coordinate systems are shown with following interpretation and given in 4x4 homogenous coordinates:

- $T_{\text{Ref}}^{\text{UTM}}$ is the transformation from the UTM-system into the virtual reference system.
- $T_{\text{IMU}}^{\text{UTM}}$ is the transformation from UTM in GPS/IMU system measuring the current flight position and orientation in RPY coordinates.
- $T_{\text{nadir}}^{\text{IMU}}$, $T_{\text{rightdir}}^{\text{IMU}}$, $T_{\text{leftdir}}^{\text{IMU}}$ are the transformations into the camera system respectively. These transformations cover the boresight angles and transforms into the principle point of the camera system.
- $T_{\text{Ortho}}^{\text{Ref}}$ defines the transformation from the reference system into the virtual camera system looking in negative z-axis. Applying orthogonal projection results the desired orthorectified image.

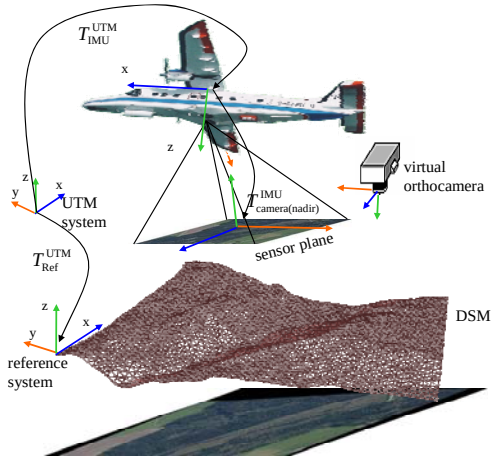

Figure 5: Arrangement in 3d. All transformations are depicted. The RPY angles are obtained by the GPS/IMU.

Some other parameters are also necessary. These are the pixel size $s = 7.21\ 10^{-6}\ m$ of camera and the number of cols and rows. Also, the focal length is given with $n = 0.0511\ m$. Up to now, we remove the radial distortion of images from the original image analytically, but we will accelerate the computation by adding an appropriate 3d-mesh to the triangled DSM. Also the distance from principle point to projection centre is necessary. These parameters as well as boresight angles are obtained from calibration. This is done on-the-fly without ground control points based on automatically matched 3-ray tie points in combination with GPS/IMU data (Kurz et al. 2007a).

4.1 Real Time Loop

As shown before, meshing DSMs is possible in real time for DSMs with resolution of up to *2 m*. In the next step, the image is tiled due to graphics texture buffer size. This varies from *2048 pixels* to *4096 pixels* according to the available GPU. Each tile is loaded and scaled with respect to the virtual image size (w_x,w_y) by applying the scaling matrix S. Then, it is

transformed into the virtual sensor plane. Following transformations are used for each tile:

$$S \cdot T_{UTM}^{\mathrm{Re}\,f} \cdot T_{IMU}^{UTM} \cdot T_{nadir}^{IMU} \cdot T_{sensorplane}^{nadir}, \qquad (1)$$

With (x,y,z) obtained from GPS and (rpy) obtained from the IMU this yields:

$$T_{IMU}^{UTM} = Trans\,(x,y,z) \cdot Rot_z(\pi/2) \cdot Rot_x(-\pi) \cdot Rot_x(-r) \cdot Rot_y(-p) \cdot Rot_z(-y) \qquad (2)$$

The last transformation is only a translation into the projection center according to the length ($n = 0.0511\ m$). A perspective projection matrix **P** is applied. It maps a 2d-point to a ray, which is intersected with the meshed surface by graphics hardware.

$$P := \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-l} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

with n = focal length equals to the near clipping plane
f = far clipping plane
r, l = right/left border of the tile to be projected
t, b = upper and lower border of the tile

The parameters are calculated according to selected tiles of the image. Figure 6 illustrates the relation. Thus, selecting the tile shown of the image to be projected results in appropriate values of *l, r, b, t*. For example, the upper left tile is projected, then *l* is set to the pixel size multiplied by half number of columns and *r* is set to zero. The other values are obtained similarly. The computation of the far clipping plane is much difficult. It depends on the current RPY angles, the boresight angle and the DSMs normals. Thus, the rotated bounding box of the underlying DSM is projected onto the *z*-axis of the sensor coordinate system. Its maximal length is applied as distance to the far clipping plane. Thus, the z-buffer size is estimated very fast. Now, the tile can be projected by OpenGL. Computation of texture coordinates is done automatically by fast graphics hardware. At next they are mapped onto the DSM.

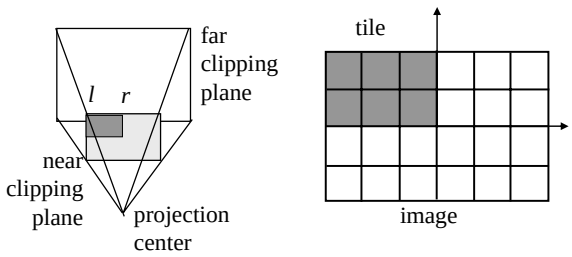

Figure 6: Projection of tiles and the corresponding OpenGL parameters, which are computed during the real time loop.

The orthorectified image can be obtained by viewing the scene from the orthocamera, which is placed according to $T_{\mathrm{Ortho}}^{\mathrm{Ref}}$. The entire scene is scanned by driving the camera in the (x,y) plane and reading the colour buffer of the result back to the CPU. To determine the number of rendering calls required during the real time loop for one image, three parameters are required: The desired resolution in pixels, the size of the resulting image, and the maximal window size available by graphics hardware available. In order to estimate the size of the resulting image prior to execution, an image with only one and the same colour value in each pixel is generated and projected with lower resolution onto the DSM. Then the colour buffer is read back and the resulting bounding box is obtained (it is an oriented bounding box, according to the orthocamera system). For calculation of necessary rendering steps, the maximised possible window size is applied. The scanning window is an axis aligned window. From that the minimal number of required image shots according to window size pixel ground resolution is computed. Figure 7 shows the coherence. For each window, corresponding tiles of the original image are selected and projected into the ortho-image.

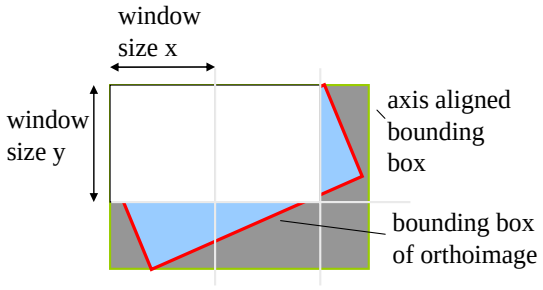

Figure 7: Scanning the high resolution image, here six rendering steps are required in the inner real time loop.

.

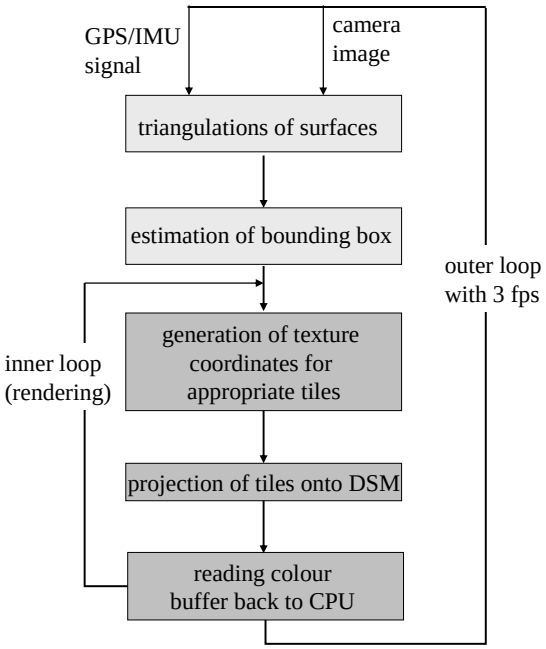

Figure 8: Processing steps to be computed within the real time loop. Each projected image is assembled to the final ortho-image.

Assembling the orthoimages is done by collecting projected tiles of parts of the image in each rendering step as illustrated in

figure 7. Images for the white tiles are obtained in each rendering step. They result in the desired orthoimage. Figure 8 depicts the two real time loops; in each computation step the inner loop corresponds to the rendering loop and the outer loop is the real time loop itself for obtaining an orthorectified image for the DLR 3K-camera system. Except from meshing the complete implementation is done in OpenGL.

4.2 Optimizations

Some optimizations concern the usage of OpenGL. Here, *Display-Lists* could be used, because the DSMs are not modified during flight. Another optimization would be the usage of some OpenGL extensions allowing textures of varying sizes. A third optimization of the orthorectification improves the distortion algorithm. Currently, we use an analytical solution. Instead of that the calibration program could generate a 3d-mesh, which is only added to the meshed DSM. Hence, no more computation time is required for radial distortion of images.

5. RESULTS

In the usual case, we will compute the triangle meshes of DSMs during the flight. For execution in real time computation times should not be too long. Thus, we have measured our algorithm for 3d-mesh triangulation. Figure 9 depicts execution times according to flight height and DSM resolution. We needed *8 ms* for loading DSM with a resolution of *25 m* and triangulating it into *6664* triangles. The triangulation was performed very fast within *0.0014 ms*. Thus, most of the execution time is caused by loading. The amount of triangles in this example corresponds to a flight height of *2000 m* in nadir. Thus, computation times for images of side looking cameras will be about 30 % higher.

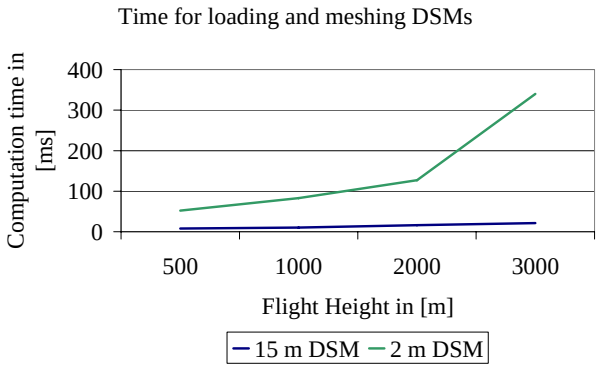

Figure 9: Computation times for loading and triangle DSMs on-the-fly in according to flight height for our DLR 3K-camera system.

Due to the high image resolution we decided to apply the *25 m* DSM in order to have enough computation time available for image loading and copying data between GPU and CPU buffer.

Another important execution time is the rendering step for a single tile, because this has to be repeated often during the real time loop. Table 2 depicts the relation between number of tiles, available texture buffer size and necessary rendering steps.

The computation times for one rendering step according to texture buffer size *512x512, 1024x124* and *2048x2048* are *17 ms*, *51 ms* and *95 ms*. The computation time was determined on a no-named Mobile on Board GPU. For reducing our execution time we will apply fast GPUs in the near future. Despite this, it is obvious that the performance of the graphics hardware suffices for orthorectification of one single image of the 3K-camera system. Thus, a frame rate of 3fps for the orthoimage process can be achieved.

Flight height	Number of rendering steps			
	buffer size of (512 x 512)		buffer size of (2048x 2048)	
	off-dir	nadir	off-nadir	nadir
1000 m	*app.* 68	*app.*60	*app.*9	*app.*6
2000 m	*app.* 80	*app.*70	*app.*12	*app.*8
3000 m	*app.* 110	*app.*80	*app.*14	*app.*10

Table 2: Relation between texture buffer size and number of iterations in the inner-loop. Assuming window buffer size and texture buffer size has the same value, also assuming a side looking angle of 35°.

For obtaining first results of our new algorithm we used images and GPS/IMUS data from a flight in southern Germany in 2007. In figure 10 the image can be seen where the radial distortion is removed. The orthorectified image tile obtained by our real time algorithm can be seen in figure 11. In comparison the orthoimage computed by (Mueller et. al. 2004) is shown in figure 11. The tile of the image is obtained within *97 ms* on a mobile no-named on board GPU.

Figure 10: Original image where distortion is already removed.

Figure 11: Result of the orthorectification of one tile.

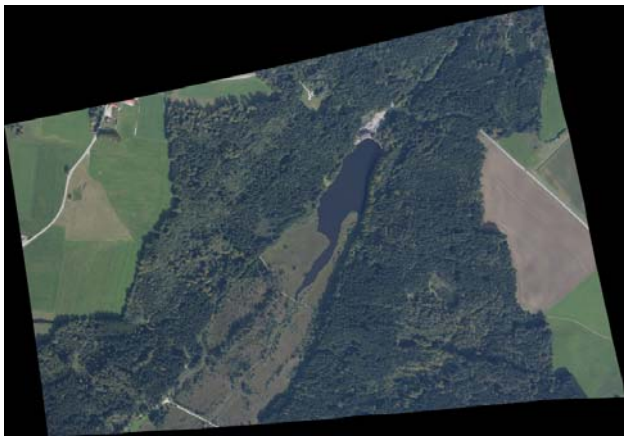

Figure 12: Orthoimage computed with the same original image and with the equal DSM. The entire image is projected according to (Mueller et. al. 2004).

6. CONCLUSION AND OUTLOOK

In the paper, we have suggested a new architecture for real time image processing in remote sensing. The architecture consists of the DLR 3K-camera system and five on board PCs. For real time applications in remote sensing e.g. traffic monitoring orthorectification of images with a rate of 3fps is required. Up-to-date analytical algorithm suffer from the high amount of data (16Mpixel) to be projected, hence a real time algorithm has been implemented which exploits graphics hardware. With this implementation orthorectifcation of images in real time has become possible. The computation times for the algorithm may change according to data set. We have evaluated our algorithm on one data set so far. In the future, we will evaluate the quality of projected images using ground control points. The accuracy of orthoprojected images has to be investigated as well. The radial distortion of images is up-to-now computed analytically. It will bee accelerated by using an appropriate 3d mesh. Altogether, we can orthorectify images in real time, which is an important first step towards real time monitoring of e.g. traffic, disaster, and mass events.

REFERENCES

Diethers, K.; Finkemeyer, B.; Kohn, N. (Diethers 2003): Middleware zur Realisierung einer offenen Steuerungshardware für hochdynamische Prozesse. In: *it-Information Technology*, (1), 2004 pp. 39-47.

Kurz, F., Müller, R., Stephani, M., Reinartz, P., Schroeder, M. (Kurz 2007 a): Calibration of a wide-angle digital camera system for near real time scenarios. In: *Heipke, C.; Jacobsen, K.; Gerke, M. [Eds.]: ISPRS Hannover Workshop 2007, High Resolution Earth Imaging for Geospatial Information, Hannover*, 2007-05-29 - 2007-06-01, ISSN 1682-1777

Kurz, F., Charmette, B., Suri, S., Rosenbaum, D., Spangler, M., Leonhardt, A., Bachleitner, M., Stätter, R., Reinartz, P. (Kurz 2007 b): Automatic traffic monitoring with an airborne wide-angle +digital camera system for estimation of travel times. In: *Stilla, U., Mayer, H., Rottensteiner, F., Heipke, C., Hinz, S. [Eds.]: The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. 36 (3/W49B), pp 83 -86

Mueller, Rupert; Lehner, M.; Mueller, Rainer; Reinartz, P.; Schroeder, M.; Vollmer, B. (Mueller 2004): A program for direct georeferencing of airborne and spaceborne line scanner images. In: *ISPRS 2004 Conference of Photogrammetry and Remote Sensing, Commission I, WG 1/5,*2004

Liu, C.-H. (Liu 2004): Fast georeferncing images through generalized photogrammetric algorithms. In: *ISPRS 2004 Conference of Photogrammetry and Remote Sensing, Commission I, WG 1/6,* 2004.

Schantz, R. E.; Loyall, J. P.; Schmidt, D. C.; Rodrigues, C.; Krishnamurthy, Y.; Pyarali, I. (Schantz 2003): Flexible and Adaptive QoS Control for Distributed Real-time and Embedded Middleware. In: *Proceedings of Middleware 2003, 4th IFIP/ACM/USENIX International Conference on Distributed Systems Platforms, June 16-20, Rio de Janeiro, Brazil*, 2003.

Woo, M.; Neider, J; Davis, T.; Shreiner, D. (Woo 1999). OpenGL Programming Guide. *Addision Wesley, Boston, 1999*.

Wright D.B.; T. Yotsumata [a,]; N. El-Sheimy Liu, C.-H: Real Tim Identification and Location of forest fire Hotspots from geo-referenced thermal images. *ISPRS 2004 Conference of Photogrammetry and Remote Sensing, Commission I, WG 1/6,* 2004.